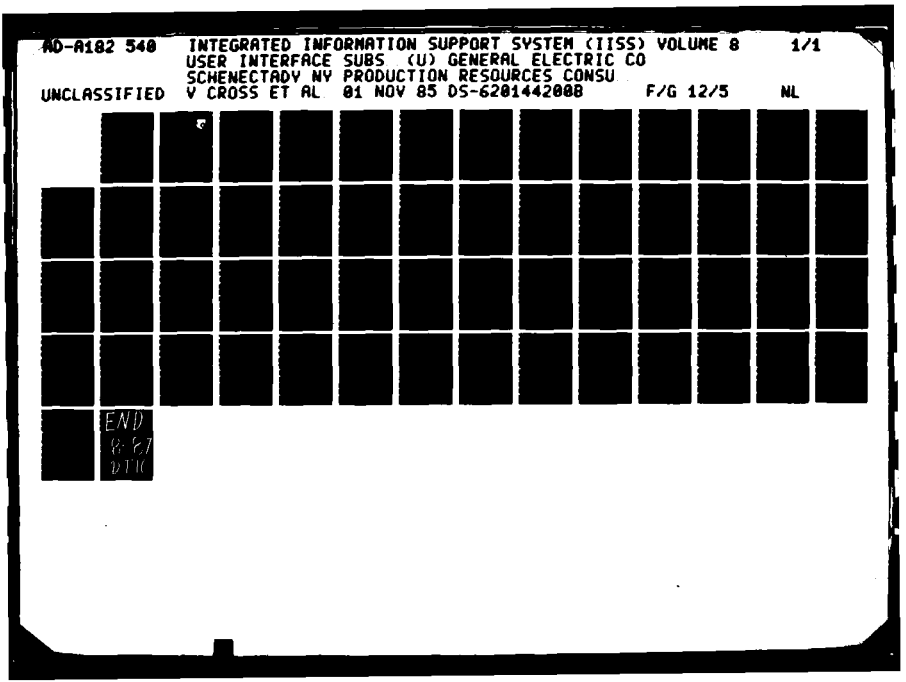
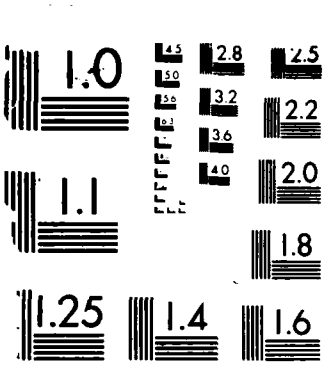

AD-A182 540 INTEGRATED INFORMATION SUPPORT SYSTEM (IISS) VOLUME 8 1/1
USER INTERFACE SUBS (U) GENERAL ELECTRIC CO
SCHENECTADY NY PRODUCTION RESOURCES CONSU
UNCLASSIFIED V CROSS ET AL 01 NOV 85 DS-6201442008 F/G 12/5 NL
END
8 87
DTIC



AFWAL-TR-86-4006
Volume VIII
Part 5

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume VIII - User Interface Subsystem
Part 5 - Form Processor Development Specification

General Electric Company
Production Resources Consulting
One River Road
Schenectady, New York 12345

Final Report for Period 22 September 1980 - 31 July 1985

November 1985

Approved for public release; distribution is unlimited.

PREPARED FOR:

MATERIALS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AFB, OH 45433-6533

NOTICE

When Government drawings specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

DAVID L. JUDSON, PROJECT MANAGER
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

5 Aug 1986
DATE

FOR THE COMMANDER:

GERALD C. SHUMAKER, BRANCH CHIEF
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

7 Aug 86
DATE

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/MLTC, W-PAFB, OH 45433 to help us maintain a current mailing list."

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

1 November 1985

REPORT DOCUMENTATION PAGE

AD-A172 540

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFWAL-TR-86-4006 Vol VIII, Part 5
6a NAME OF PERFORMING ORGANIZATION General Electric Company Production Resources Consulting	6b. OFFICE SYMBOL (If applicable)
6c. ADDRESS (City, State and ZIP Code) 1 River Road Schenectady, NY 12345	7a. NAME OF MONITORING ORGANIZATION AFWAL/MLTC
	7b. ADDRESS (City, State and ZIP Code) WPAFB, OH 45433-6533
8a NAME OF FUNDING/SPONSORING ORGANIZATION Materials Laboratory Air Force Systems Command, USAF	8b. OFFICE SYMBOL (If applicable) AFWAL/MLTC
	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-80-C-5155
8c ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, Ohio 45433	10 SOURCE OF FUNDING NOS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT NO
78011F	7500	62	01

11 TITLE (Include Security Classification)
(See Reverse)

12. PERSONAL AUTHOR(S) Cross, Valerie and Morenc, Carol and Robie, Penny

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Yr., Mo., Day)	15 PAGE COUNT
Final Technical Report	22 Sept 1980 - 31 July 1985	1985 November	55

16 SUPPLEMENTARY NOTATION
ICAM Project Priority 6201

The computer software contained herein are theoretical and or references that in no way reflect Air Force-owned or -developed computer software.

17 COSATI CODES		
FIELD	GROUP	SUB GR
1308	0905	

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19 ABSTRACT

This specification establishes the development, test and qualification requirements of computer program identified as the Form Processor (FP). The FP is used to permit an application to send receive data on the formatted screen without having the application program know all the characteristics of the particular terminal being used. It is used in conjunction with the Virtual Terminal.

When used in the IISS environment, the application programs use the Application Interface (AI) to create the appropriate messages which are sent to the User Interface Monitor (UIM) of the FP by way of the Network Transaction Manager (NTM). These messages are then decoded into the appropriate Form Processor routines. The routines needing to input output to the terminal then translate the request into the correct VT commands. If the FP is being used on a one-computer system, the application program would be directly interfacing with the FP callable routines instead of the AI which generates the network formatted message. The FP also handles window management processing which requires the FP to be in window manager mode or to be servicing the window management form.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS ☐	Unclassified

22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE NUMBER (Include Area Code)	22c OFFICE SYMBOL
David L. Judson	513 255 6976	AFWAL MLTC

DD FORM 1473, 83 APR

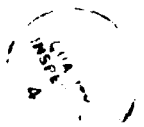
EDITION OF 1 JAN 73 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

11. Title

Integrated Information Support System (IISS)
Vol VIII - User Interface Subsystem
Part 5 - Form Processor Development Specification



PREFACE

This development specification covers the work performed under Air Force Contract F33615-80-C-5155 (ICAM Project 6201). This contract is sponsored by the Materials Laboratory, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Gerald C. Shumaker, ICAM Program Manager, Manufacturing Technology Division, through Project Manager, Mr. David Judson. The Prime Contractor was Production Resources Consulting of the General Electric Company, Schenectady, New York, under the direction of Mr. Alan Rubenstein. The General Electric Project Manager was Mr. Myron Hurlbut of Industrial Automation Systems Department, Albany, New York.

Certain work aimed at improving Test Bed Technology has been performed by other contracts with Project 6201 performing integrating functions. This work consisted of enhancements to Test Bed software and establishment and operation of Test Bed hardware and communications for developers and other users. Documentation relating to the Test Bed from all of these contractors and projects have been integrated under Project 6201 for publication and treatment as an integrated set of documents. The particular contributors to each document are noted on the Report Documentation Page (DD1473). A listing and description of the entire project documentation system and how they are related is contained in document FTR620100001, Project Overview.

The subcontractors and their contributing activities were as follows:

TASK 4.2

Subcontractors	Role
Boeing Military Aircraft Company (BMAC)	Reviewer.
D. Appleton Company (DACOM)	Responsible for IDEF support, state-of-the-art literature search.
General Dynamics/ Ft. Worth	Responsible for factory view function and information models.

Subcontractors	Role
Illinois Institute of Technology	Responsible for factory view function research (IITRI) and information models of small and medium-size business.
North American Rockwell	Reviewer.
Northrop Corporation	Responsible for factory view function and information models.
Pritsker and Associates	Responsible for IDEF2 support.
SofTech	Responsible for IDEF0 support.

TASKS 4.3 - 4.9 (TEST BED)

Subcontractors	Role
Boeing Military Aircraft Company (BMAC)	Responsible for consultation on applications of the technology and on IBM computer technology.
Computer Technology Associates (CTA)	Assisted in the areas of communications systems, system design and integration methodology, and design of the Network Transaction Manager.
Control Data Corporation (CDC)	Responsible for the Common Data Model (CDM) implementation and part of the CDM design (shared with DACOM).
D. Appleton Company (DACOM)	Responsible for the overall CDM Subsystem design integration and test plan, as well as part of the design of the CDM (shared with CDC). DACOM also developed the Integration Methodology and did the schema mappings for the Application Subsystems.

Subcontractors	Role
Digital Equipment Corporation (DEC)	Consulting and support of the performance testing and on DEC software and computer systems operation.
McDonnell Douglas Automation Company (McAuto)	Responsible for the support and enhancements to the Network Transaction Manager Subsystem during 1984/1985 period.
On-Line Software International (OSI)	Responsible for programming the Communications Subsystem on the IBM and for consulting on the IBM.
Rath and Strong Systems Products (RSSP) (In 1985 became McCormack & Dodge)	Responsible for assistance in the implementation and use of the MRP II package (PIOS) that they supplied.
SofTech, Inc.	Responsible for the design and implementation of the Network Transaction Manager (NTM) in 1981/1984 period.
Software Performance Engineering (SPE)	Responsible for directing the work on performance evaluation and analysis.
Structural Dynamics Research Corporation (SDRC)	Responsible for the User Interface and Virtual Terminal Interface Subsystems.

Other prime contractors under other projects who have contributed to Test Bed Technology, their contributing activities and responsible projects are as follows:

Contractors	ICAM Project	Contributing Activities
Boeing Military Aircraft Company (BMAC)	1701, 2201, 2202	Enhancements for IBM node use. Technology Transfer to Integrated Sheet Metal Center (ISMC).

Contractors	ICAM Project	Contributing Activities
Control Data Corporation (CDC)	1502, 1701	IISS enhancements to Common Data Model Processor (CDMP).
D. Appleton Company (DACOM)	1502	IISS enhancements to Integration Methodology.
General Electric	1502	Operation of the Test Bed and communications equipment.
Hughes Aircraft Company (HAC)	1701	Test Bed enhancements.
Structural Dynamics Research Corporation (SDRC)	1502, 1701, 1703	IISS enhancements to User Interface/Virtual Terminal Interface (UI/VTI).
Systran	1502	Test Bed enhancements. Operation of Test Bed.

TABLE OF CONTENTS

APPENDICES

FIGURES

SECTION 1

SCOPE

1.1 Identification

This specification establishes the development, test and qualification requirements of a computer program identified as the Form Processor, referred to as the FP. The FP is one configuration item of the Integrated Information Support System (IISS) User Interface (UI) and consists of the User Interface Monitor (UIM), the form processor callable routines, and the window management processing.

1.2 Functional Summary

One of the objectives of the IISS testbed is to allow applications to be run from a wide variety of terminals using formatted screens for input and output of application data. Instead of the application programs having to contain terminal dependent code to send/receive formatted screens to/from various types of terminals and to perform terminal control functions, the program may use the set of callable execution time routines of the FP.

The major functions of the FP are:

1. Opening and displaying a form, a template defining fields and their attributes.

2. Placing data into a form and/or into a form message line.

3. Sending the form to the terminal.

4. Reading the data from the terminal.

5. Stacking/replacing forms currently open for the application program.

6. IISS Logon Processing.

7. NTM Message Processing.

8. Window Management Processing.

SECTION 2

DOCUMENTS

2.1 Reference Documents

[1] General Electric Co., ICAM Integrated Support System (IISS) Test Bed System Design Specification (Draft), 7 Feb 83, SDS620140000.

[2] Structural Dynamics Research Corporation, IISS Form Processor User Manual, 1 November 1985.

[3] ICAM Documentation Standards, 15 September 1983, IDS150120000C.

[4] Structural Dynamics Research Corporation, Report Writer Development Specification, DS 620144501 , 1 November 1985.

[5] Structural Dynamics Research Corporation, Rapid Application Generator Development Specification, DS 620144502 , 1 November 1985.

[6] Structural Dynamics Research Corporation, Text Editor Development Specification, DS 620144600B, 1 November 1985.

[7] Structural Dynamics Research Corporation, Form Processor Development Specification, DS 620144200B, 1 November 1985.

[8] Structural Dynamics Research Corporation, Application Interface Development Specification, DS 620144700 , 1 November 1985.

[9] Structural Dynamics Research Corporation, Forms Language Compiler Development Specification, DS 620144401B, 1 November 1985.

[10] Structural Dynamics Research Corporation, Forms Driven Form Editor Development Specification, DS 620144402B, 1 November 1985.

[11] Structural Dynamics Research Corporation, User Interface Services Development Specification, DS 620144100B, 1 November 1985.

[12] Structural Dynamics Research Corporation, Virtual Terminal Development Specification, DS 620144300B, 1 November 1985.

2.2 Terms and Abbreviations

American Standard Code for Information Interchange: (ASCII), the character set defined by ANSI X3.4 and used by most computer vendors.

Application Interface: (AI), subset of the IISS User Interface that consists of the callable routines that are linked with applications that use the Form Processor or Virtual Terminal. The AI enables applications to be hosted on computers other than the host of the User Interface.

Application Process: (AP), a cohesive unit of software that can be initiated as a unit to perform some function or functions.

Attribute: field characteristic such as blinking, highlighted, black, etc. and various other combinations. Background attributes are defined for forms or windows only. Foreground attributes are defined for items. Attributes may be permanent, i.e., they remain the same unless changed by the application program, or they may be temporary, i.e., they remain in effect until the window is redisplayed.

Common Data Model: (CDM), IISS subsystem that describes common data application process formats, form definitions, etc. of the IISS and includes conceptual schema, external schemas, internal schemas, and schema transformation operators.

Computer Program Configuration Item: (CPCI), an aggregation of computer programs or any of their discrete portions, which satisfies an end-use function.

Conceptual Schema: (CS), the standard definition used for all data in the CDM. It is based on IDEF1 information modelling.

Current Cursor Position: the position of the cursor before an edit command or function is issued in the text editor.

Cursor Position: the position of the cursor after any command is issued.

Device Drivers: (DD), software modules written to handle I/O for a specific kind of terminal. The modules map terminal specific commands and data to a neutral format. Device Drivers are part of the UI Virtual Terminal.

Display List: is similar to the open list, except that it contains only those forms that have been added to the screen and are currently displayed on the screen.

Display Size: the number of lines used in the edit area.

Extended Binary Coded Decimal Interchange Code: (EBCDIC), the character set used by a few computer vendors (notably IBM) instead of ASCII.

External Schema: (ES), an application's view of the CDM's conceptual schema.

Field: two-dimensional space on a terminal screen.

Field Pointer: indicates the ITEM which contains the current cursor position.

Form: structured view which may be imposed on windows or other forms. A form is composed of fields. These fields may be defined as forms, items, and windows.

Form Definition: (FD), forms definition language after compilation. It is read at runtime by the Form Processor.

Forms Definition Language: (FDL), the language in which electronic forms are defined.

Forms Driven Form Editor: (FDFE), subset of the FE which consists of a forms driven application used to create Form Definition files interactively.

Form Editor: (FE), subset of the IISS User Interface that is used to create definitions of forms. The FE consists of the Forms Driven Form Editor and the Forms Language Compiler.

Form Hierarchy: a graphic representation of the way in which forms, items and windows are related to their parent form.

Forms Language Compiler: (FLAN), subset of the FE that consists of a batch process that accepts a series of forms definition language statements and produces form definition files as output.

Form Processor: (FP), subset of the IISS User Interface that consists of a set of callable execution time routines available to an application program for form processing.

Form Processor Text Editor: (FPTE), subset of the Form Processor that consists of software modules that provide text editing capabilities to all users of applications that use the Form Processor.

Integrated Information Support System: (IISS), a test computing environment used to investigate, demonstrate and test the concepts of information management and information integration in the context of Aerospace Manufacturing. The IISS addresses the problems of integration of data resident on heterogeneous data bases supported by heterogeneous computers interconnected via a Local Area Network.

Item: non-decomposable area of a form in which hard-coded descriptive text may be placed and the only defined areas where user data may be input/output.

Logical Device: a conceptual device which to an application is indistinguishable from a physical device and is then mapped to part or all of a physical device.

Message: descriptive text which may be returned in the standard message line on the terminal screen. They are used to warn of errors or provide other user information.

Message Line: a line on the terminal screen that is used to display messages.

Network Transaction Manager: (NTM), IISS subsystem that performs the coordination, communication and housekeeping functions required to integrate the Application Processes and System Services resident on the various hosts into a cohesive system.

Open List: a list of all the forms that have been and are currently open for an application process

Operating System: (OS), software supplied with a computer which allows it to supervise its own operations and manage access to hardware facilities such as memory and peripherals.

Page: instance of forms in windows that are created whenever a form is added to a window.

Paging and Scrolling: a method which allows a form to contain more data than can be displayed with provisions for viewing any portion of the data buffer.

Physical Device: a hardware terminal.

Presentation Schema: (PS), may be equivalent to a form. It is the view presented to the user of the application.

Previous Cursor Position: the position of the cursor when the previous edit command was issued.

Qualified Name: the name of a form, item or window preceded by the hierarchy path so that it is uniquely identified.

Report Definition Language: an extension of the Forms Definition Language that includes retrieval and calculation of database information and is used to define reports.

Subform: a form that is used within another form.

User Data: data which is either input by the user or output by the application programs to items.

User Interface: (UI), IISS subsystem that controls the user's terminal and interfaces with the rest of the system. The UI consists of two major subsystems: the User Interface Development System (UIDS) and the User Interface Management System (UIMS).

User Interface Development System: (UIDS), collection of IISS User Interface subsystems that are used by applications programmers as they develop IISS applications. The UIDS includes the Form Editor and the Application Generator.

User Interface Management System: (UIMS), the runtime UI. It consists of the Form Processor, Virtual Terminal, Application Interface, the User Interface Services and the Text Editor.

User Interface Monitor: (UIM), part of the Form Processor that handles messaging between the NTM and the UI. It also provides authorization checks and initiates applications.

User Interface/Virtual Terminal Interface: (UI/VTI), another name for the User Interface.

Virtual Terminal: (VT), subset of the IISS User Interface that performs the interfacing between different terminals and the UI. This is done by defining a specific set of terminal features and protocols which must be supported by the UI software which constitutes the virtual terminal definition. Specific terminals are then mapped against the virtual terminal software by specific software modules written for each type of real terminal supported.

Virtual Terminal Interface: (VTI), the callable interface to the VT.

Window: dynamic area of a terminal screen on which predefined forms may be placed at run time

Window Manager: a facility which allows the following to be manipulated: size and location of windows, the device on which an application is running, the position of a form within a window. It is part of the Form Processor

SECTION 3

REQUIREMENTS

This section includes functional and performance requirements for the FP. In addition, the FP interfaces to other IISS Computer Program Configuration Item's (CPCI's) are defined.

3.1 Computer Program Definition

The Form Processor is used to permit an application to send/receive data on a formatted screen without having the application program know all the characteristics of the particular terminal being used. It is used in conjunction with the Virtual Terminal.

When used in the IISS environment, the application programs use the Application Interface (AI) to create the appropriate messages which are sent to the User Interface Monitor (UIM) of the Form Processor by way of the Network Transaction Manager (NTM). These messages are then decoded into the appropriate Form Processor routines. The routines needing to input/output to the terminal then translate the request into the correct Virtual Terminal commands. If the Form Processor is being used on a one computer system, the application program would be directly interfacing with the FP callable routines instead of the Application Interface (AI) which generates the network formatted message. The Form Processor also handles window management processing which requires the Form Processor to be in window manager mode or to be servicing the window management form.

3.1.1 System Capacities

The system capacities required to run the FP should be defined in terms of additional memory requirements for application programs using the Form Processor and in terms of additional timing overhead for using the FP to send/receive data to/from the terminal. The limit to the number of forms being handled by the FP is restricted by the available process memory.

3.1.2 Interface Requirements

The FP interfaces with the NTM through its UIM, with the AI

through its UIM and callable routines, and with the VT through the VT Form Processor callable routines. In addition, two different methods of interfacing to an AP are supported: stand alone, when the AP may link directly to the FP and no NTM is being used, or the IISS environment, when the AP may link to the Application Interface (AI) routines. In either environment, the application programs use the exact same interface to the FP. Linking to the FP directly is simpler and more efficient than using the AI but does not support the use of an NTM; direct linking is most useful for testing purposes.

The UIM part of the FP only exists when the NTM is being used. The UIM receives the message formatted by the AI and translates it into the appropriate FP callable routine to permit the sending or receiving of the forms. The Form Processor routines then interface with the VT by translating an application request into the appropriate VT command when input/output is necessary.

3.1.2.1 Interface Block Diagram

The structure of the FP interfaces is shown in Figure 3-1.

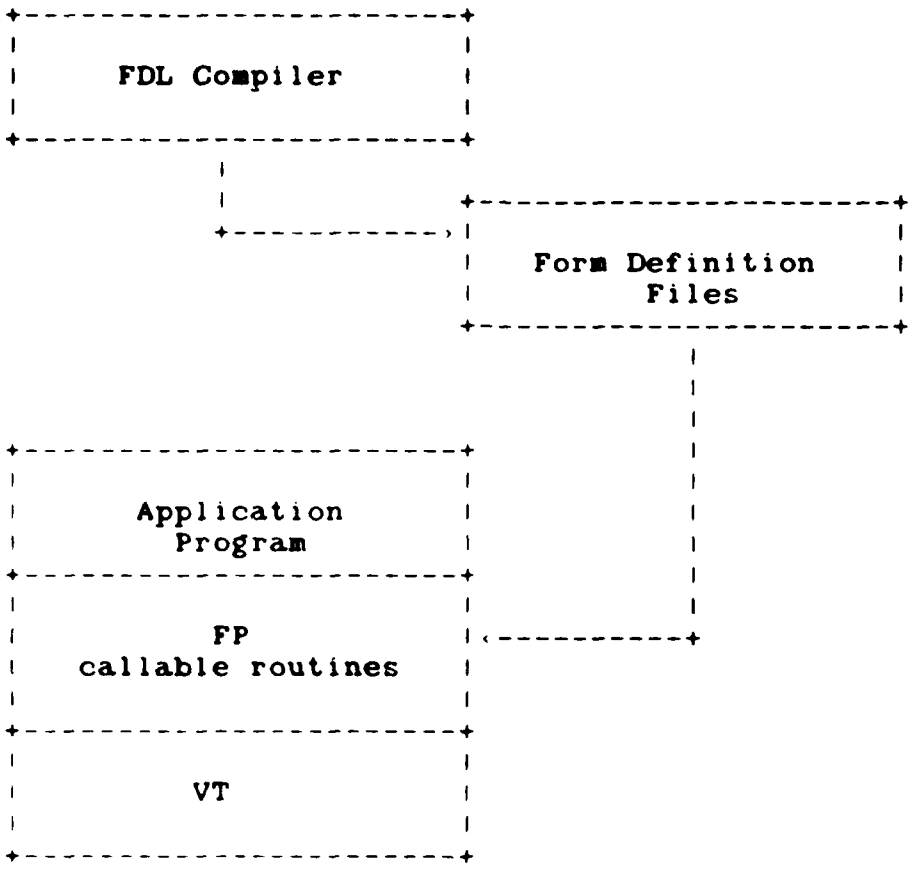


Figure 3-1a FP Stand Alone (non IISS environment)

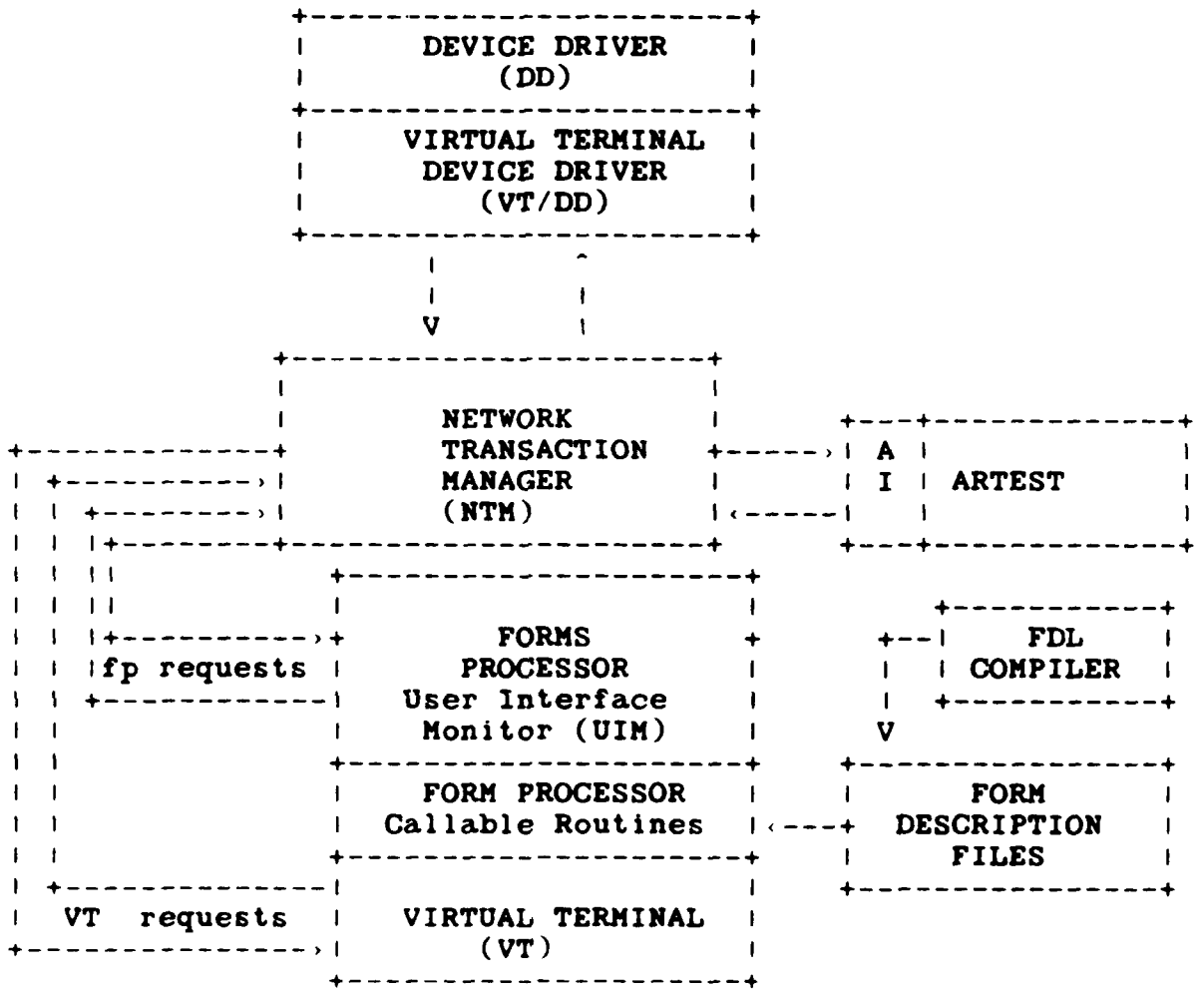


Figure 3-1b FP in IISS environment

3.1.2.2 Detailed Interface Definition

3.1.2.2.1 Application

The Form Processor interface for IISS applications is the set of callable execution time routines available for an application program for form processing. These routines are defined in the IISS Form Processor User's Manual. The FP routines allow application programs and their users to communicate through predefined forms on a terminal. Again the

application may directly interface with the FP through the FP routines or through the AI routines. In either case, the calling sequence is exactly the same.

The typical order of events in an application program using these routines is:

1. Initialize form processor.
2. Open forms.
3. Add forms to windows.
4. Put data in forms.
5. Put data in message line.
6. Send screen to terminal.
7. Read data from terminal.

When done processing, the forms should be closed, then go to step #10.

8. Remove pages that will not be displayed.
9. Replace forms if necessary.

To continue return to step #3.

10. Terminate form processor.

An application program that is to be integrated into the IISS and that uses the FP routines must be written in the format that includes initialization (INITFP) and termination (TERMFP) calls at the beginning and end of the Procedure Division (COBOL) as shown in Figure 3-2.

```
                        AP PROGRAM

  PROCEDURE DIVISION.

        CALL "INITFP".
            .
            .
            .

        AP CODE
            .
            .
            .

        CALL "TERMFP".
  EXIT-PROGRAM.
```

Figure 3-2 COBOL Procedure Division

3.1.2.2.2 VT

The FP interfaces with the VT by means of using the VT callable routines. Use of these routines is only necessary when initializing the FP (INITFP), terminating the FP (TERMFP), and outputting data to the terminal and receiving data from the terminal (OISCR, OUTSCR).

3.1.2.2.3 FDL

The FP interfaces with the FDL by use of the Forms Definition File. This file contains the binary definition of the forms that the FP may use. It simply reads in a form by form name once an Open Form request is issued for a given form.

3.2 Detailed Functional Requirements

3.2.1 General Concepts

The following sections describe some concepts that are necessary to understand before specifying the detailed functional requirements of the FP.

3.2.1.1 User Diagram Examples

The diagram Figure 3-3 illustrates the Form Processor data terminology and relates these data types to one another with respect to a user's view of a terminal.

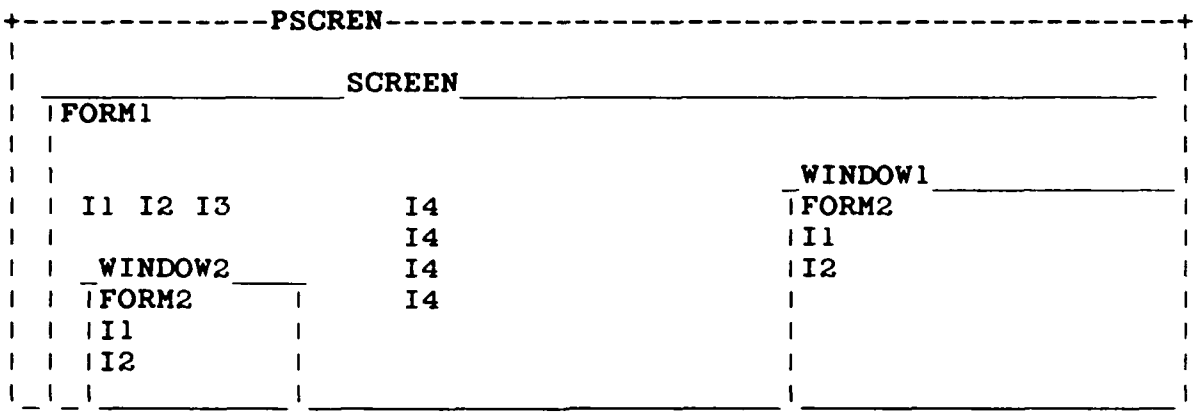


Figure 3-3 Data Type Hierarchy

The instances of the various data types in Figure 3-3 can be identified by using qualified names. The following examples illustrate this.

> "PSCREN.SCREEN.FORM1.WINDOW1.FORM2.I2;" identifies the item named I2 in the form named FORM2 on the window named WINDOW1 on the form named FORM1 on the window named SCREEN.
>
> "PSCREN.SCREEN‹99›.FORM1.WINDOW1‹98›.FORM2.I1;" identifies the item named I1 in the form named FORM2 on page 98 of the window named WINDOW1 on the form named FORM1 on page 99 of the window named SCREEN.

In addition to qualified names that identify instances of a data type, simple names may be used to identify a form. For example, FORM1, identifies a particular structured view but does not necessarily imply an instance of a page.

3.2.1.2 Paging and Scrolling

Paging and scrolling refer to the capability of displaying only a portion of the total information associated with a particular field. To implement paging and scrolling, it is

necessary to allow fields to be array structures, i. e. a field may be an array of other fields or even an array of arrays.

The difference between scrolling and paging is defined as such: scrolling occurs when the cursor is moved by one element of the array and paging occurs when the cursor is moved by the number of elements currently displayed on the screen. Scrolling and paging may be performed both horizontally and vertically. Scrolling and paging may only take place if the elements being scrolled or paged are homogeneous. For example, if an array of windows existed and each window did not contain the same form, then scrolling and/or paging would not be permitted on this array. When a user asks for scrolling or paging in a specified direction, the form processor searches the display list backwards from the current cursor position to determine which field may be scrolled/paged in the requested direction.

Each array structure has associated with it two pieces of information which determined what part of the array is currently displayed on the screen: the first element of the array to display and the number of elements of the array to display. By using this information the form processor is able to display the correct portion of the array as the scrolling or paging is requested.

3.2.1.3 Help Forms

Any input item defined for a form may have a help form or help message associated with it. A help form is defined as a form itself but is used only when the "HELP" key is pressed while positioned at the particular field. Upon pressing the "HELP" key, the associated help form or message is displayed to the user.

3.2.1.4 Terminal Within a Terminal

The concept of terminal within a terminal refers to the use of windows. When the application program displays the screen and waits for input from the screen, it provides the name of the window to be used for inputting data. Everything on the screen outside this window is automatically guarded except for the standard message line. This protection means the user may only enter data in the specified window.

3.2.1.5 Reserved Keys

Certain virtual terminal function keys are reserved by the FP to have a specific meaning. See the Terminal Operator's Guide. The terminal mapping charts in Appendix B of the Terminal Operator's Guide list the actual keys that map to the Form Processor functions keys.

3.2.1.6 ITEM Values

Every item defined for a form has a value associated with it. This can be a constant value or it may depend on the value of another field or fields. If it does depend on another field, it is referred to as a calculated field and its value will be recomputed whenever a field it depends on is changed. The value of a field can be changed in three ways: by the application program calling PDATA, by the user entering data from the terminal (if it is an input field), and by the Form Processor recomputing it when one of the fields it depends on changes (note that constant values are never recomputed).

The order in which field values are recomputed is undefined and a field is only recomputed once per display so using one calculated field in another field's value is not supported.

3.2.1.7 Window Management

3.2.1.7.1 User, Device, and Application Relationships

Window Management processing should allow the following mappings:

1) A single user mapped to one logical device and one application.

2) A single user mapped to multiple devices and one application.

3) A single user mapped to one device and multiple applications.

4) An application without a user mapped to multiple devices.

Appendix A provides figures which demonstrate these mappings.

3.2.1.7.2 Application Functionality

An application needs to be able to control the allocation of logical devices. It may do so by using the following Form Processor routines:

Inquire Logical Device (INQLDV) - Allows the application to determine the logical device number it is currently using.

Open Logical Device (OPNLDV) - Provides the application with a new device.

Change Logical Device (CHGLDV) - Allows the application to change which device it is using.

Close Logical Device (CLSLDV) - Allows the application to remove this device when it no longer is needed.

3.2.1.7.3 Window Manager Mode

Window Manager Mode is another mode of the Form Processor. It allows a user to manipulate the windows that are displayed on the screen while a single application or several applications are running. The following functions which are described in the Terminal Operator's Guide are provided for the user when in Window Manager Mode:

3.2.1.7.3.1 Selection

Selecting a window allow the user to change the size and location and scroll information. This also puts the window on the top of the stack so that the user can view it completely. All the following Window Manager mode funcitons require that the window be selected first.

To select a window, the user positions the cursor within the window and presses the SELECT function key.

3.2.1.7.3.2 Scrolling

The data being displayed in a window does not have to fit within the window completely; therefore scrolling keys are necessary to move the display of data around to position different parts of the data in the window.

Scrolling may be done up, down, left, or right by positioning the cursor on the data and pressing the appropriate

scroll key. That position on the data is then moved to the edge of the window in the appropriate direction.

3.2.1.7.3.3 Size

After selecting the window, the cursor is positioned by the user to where the user wants the lower right-hand corner of the window to be, and then the SIZE function key is pressed.

3.2.1.7.3.4 Location

To move the selected window, the user positions the cursor to where the user wants the upper left-hand corner of the window to be and presses the LOCATION function key.

3.2.1.7.3.5 Restore

A user restores a selected window to its previous position on the stack by pressing the RESTORE function key.

3.2.1.7.3.6 Function

To display the function screen so a user can request a different function to begin processing, the user presses the FUNCTION key.

3.2.1.7.3.7 Application

To select the initial window of the application, the user positions the cursor anywhere within the application's display area and presses the APPLICATION function key.

3.2.1.7.3.8 Home View

To return a form that has previously been scrolled back to its original position in the window, the user presses the HOME VIEW function key.

3.2.1.7.4 Window Manager Information Form

Some operations that can be performed on windows cannot be done in keypad mode, such as changing the precedence of the applications being displayed or changing the device where the window is being displayed. For these operations, the WINDOW function may be entered on the IISS Function form. A figure of this form may be found in Appendix B. This form may be used by the user to perform the following operations:

3.2.1.7.4.1 Review Windows

The information displayed includes: the device on which the application is displayed, the order in which applications are stacked on the screen, and the location, size and viewport offset of the windows. Viewport is the offset from the upper left-hand corner of the logical screen to the upper left-hand corner of the physical screen.

3.2.1.7.4.2 Change Device

The device name, as used and recognized by the NTM, of each application that is running on the terminal is displayed on the form. The user can change this name and move the initial window of any application to any other device that is hardwired to the system.

3.2.1.7.4.3 Change Window Size

The size of all windows is displayed on the form. The user can change the size of any window. This includes giving dimension to a window with 0,0 size that has been hidden and so restore its visibility.

3.2.1.7.4.4 Change Location

The location of the top left row and column of the window relative to the upper left corner of the parent window is displayed. The user can change these values and so change the relative position of the window.

3.2.1.7.4.5 Change Priority

The order in which the windows are stacked on the screen is displayed on the form as the priority number of the window. Thus the last application to be initiated has a priority number of 1 and is on top of the stack of initial windows and is totally visible. The user can change this number to give any application top priority. This has the same effect as selecting a window interactively using the SELECT function of the Window Manger Mode.

3.2.1.8 User Interface Monitor Processing

The User Interface Monitor of the Form Processor handles presenting and processing the IISS Logon Screen and the IISS Function Form. It also handles sending and receiving NTM

messages from/to the IISS applications for their Form Processor calls and from/to the Virtual Terminal Device Driver.

3.2.2 External Data View

3.2.2.1 IDEF1 Model

The IDEF1 model translates the data types that are relevant from an applications requirement for forms processing into a conceptual view. The following model in Figure 3-4a defines the entities and relationships for the FP window management data. The model in Figure 3-4b defines the entities and relationships for the FP form data. The following entities

PS_ES_Map,
PS_Store_data_type,
PS_Control_data type,
PS_Control_domain,
PS_Store_domain, and
(data item)

are future considerations for CDM integration.

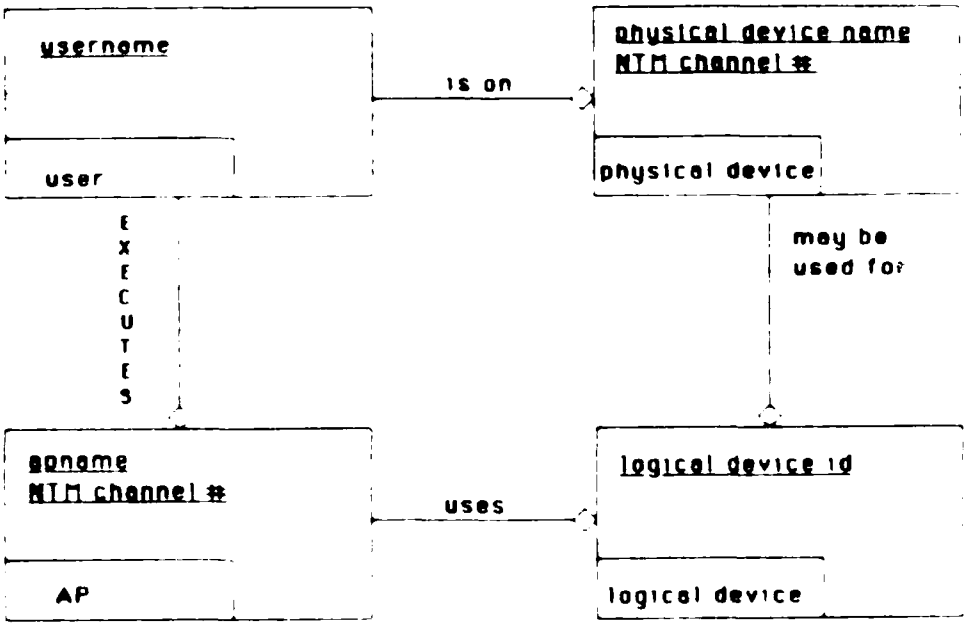


Figure 3-4a IDEF1 Model of Form Processor Window Management Data

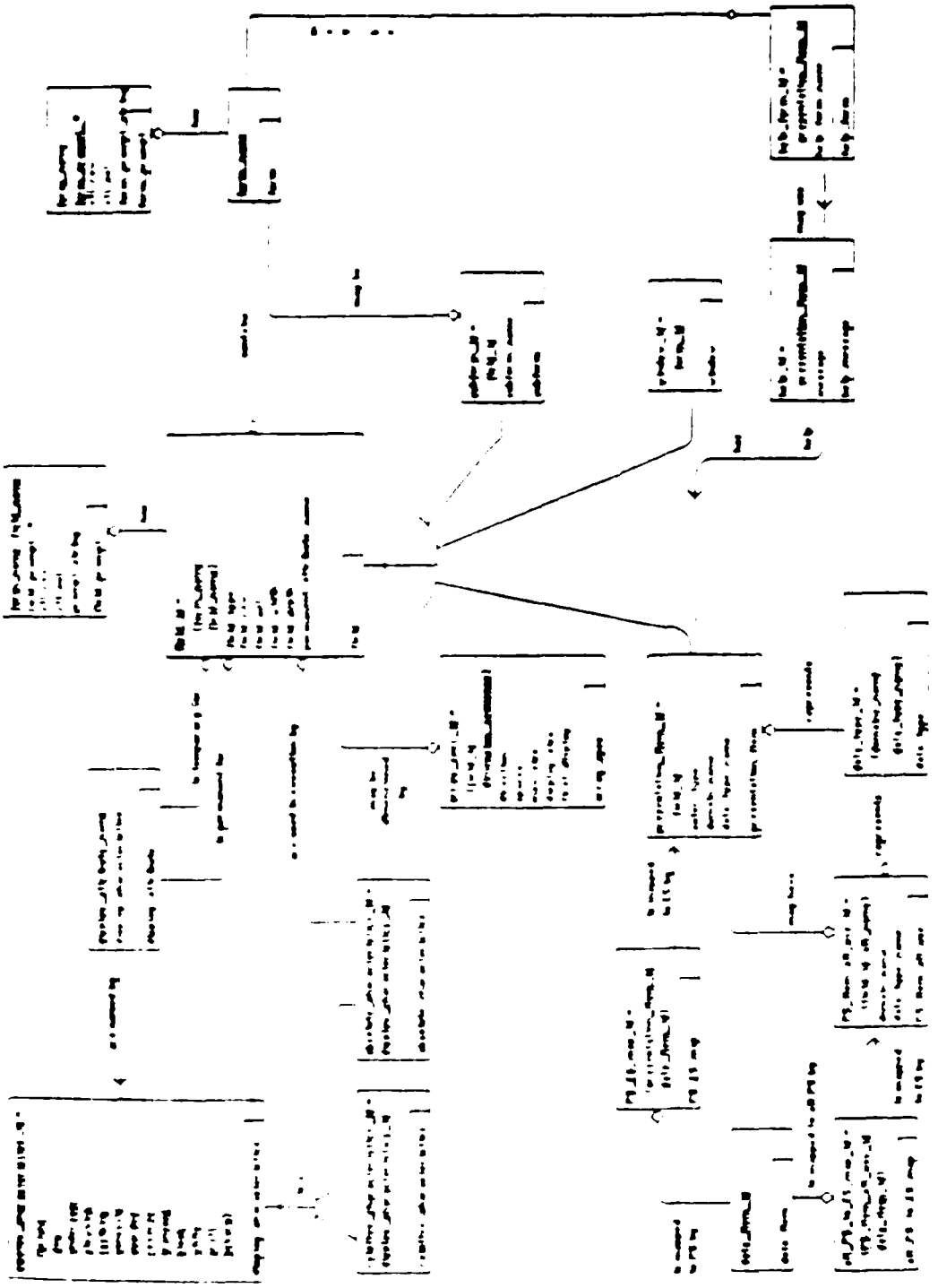


Figure 3-4b IDEF1 Model of Form Processor Form Data

3.2.2.2 Application Data Schema

The following table shows the relationships between the types of data and the functions on them. This table is defined in terms of the application views data usage. Internal changes to these data resulting from the use of these functions is described by the internal data schema.

The following abbreviations are used for purposes of this table:

n - name
pn - path name
- page number
v - value
a - array (indicates this may be an array of the specified data type)

I(n) - Input the name.
I(pn) - Input the path name.
I(#) - Input page number.
I(v) - Input the value.
O(l) - Output the length.
O(n) - Output the name.
O(pn) - Output the path name.
O(#) - Ouput page number.
O(v) - Output the value of the instance of the data type.
I(pn)* - For a function which may input different types of data, one of these types is Input to the function

The Form Processor routines CHGLDV, CLSLDV, INQLDV, and OPNLDV are not in this table because they operate on logical devices only.

Function	WINDOW	PAGE	FORM	ITEM	USER DATA	MESSAGE	ATTRIBUTE
ADDFRM	I(pn)	O(#)	I(n)				
CLSFRM			I(n)				
GDATA			I(pna)*	I(pna)*	O(v)		
GDATLN			I(pna)*	I(pna)*	O(l)		
GETATT				I(pn)			O(n)
GETBAK	I(pn)*		I(pn)*				O(n)
GPAGE	I(pn)	I(#)	O(n)				
GWINDO	I(n)	O(#)					
INITFP							
OISCR	I(pn)						
OPNFRM			I(n)				
OUTSCR	I(pn)						
PDATA			I(pna)*	I(pna)*	I(v)		
PARFQN	I(pn)* O(n)*		I(pna)* O(n)*	I(pna)* O(n)*			
PMSGLC						I(n)	

-CONTINUED-

Function	WINDOW	PAGE	FORM	ITEM	USER DATA	MESSAGE	ATTRIBUTE
PMSGLS						I(v)	
PUTATT				I(pn)			I(n)
PUTBAK	I(pn)*		I(pn)*				I(n)
PUTCUR	I(pn)*		I(pn)*	I(pn)*			
RMVPAG	I(pn)	I(*)					
RPLFRM	I(pn)	I(*)	I(n)				
TERMFP							

Figure 3-5 Table of Application Data Changes

3.2.3 Internal Data View

The following sections describe the internal data that is maintained by the FP for the application program. The representation of this internal data does not necessarily imply its implementation but suggests the relationships between the various types of data and the operations performed on the data. Figure 3-6 shows the basic elements of the Form Processor.

3.2.3.1 Open List

The Open List is a list of all the forms that are currently open for an application process. Figure 3-8 shows the Open List after FP initialization. Figure 3-9 shows the Open List after opening form F1 containing Form F2. Figure 3-10 shows the Open List after form F1 is added to the window SCREEN. Note the links in the next use field between the Open List and the display list.

3.2.3.2 Display List

The Display list is similar to the open list except it contains all forms that have been added to the screen and are currently displayed on the screen. Figure 3-8 shows the Display List after FP initialization. Figure 3-11 shows the Display List after the form F1 has been added to the window SCREEN. Note the links in the open pointer field between the Open List and the Display List after the form F1 is added to the window SCREEN.

3.2.3.3 Window Management Lists

Window management processing is based on operations on the following lists. Figure 3-7 illustrates elements of these lists.

3.2.3.3.1 User List

This list contains all User Interface users. Each user list element has an application list and a physical device list associated with it.

3.2.3.3.2 Physical Device List

This list contains all User Interface physical devices. Each physical device list element has associated with it the user and a list of all logical devices for the user.

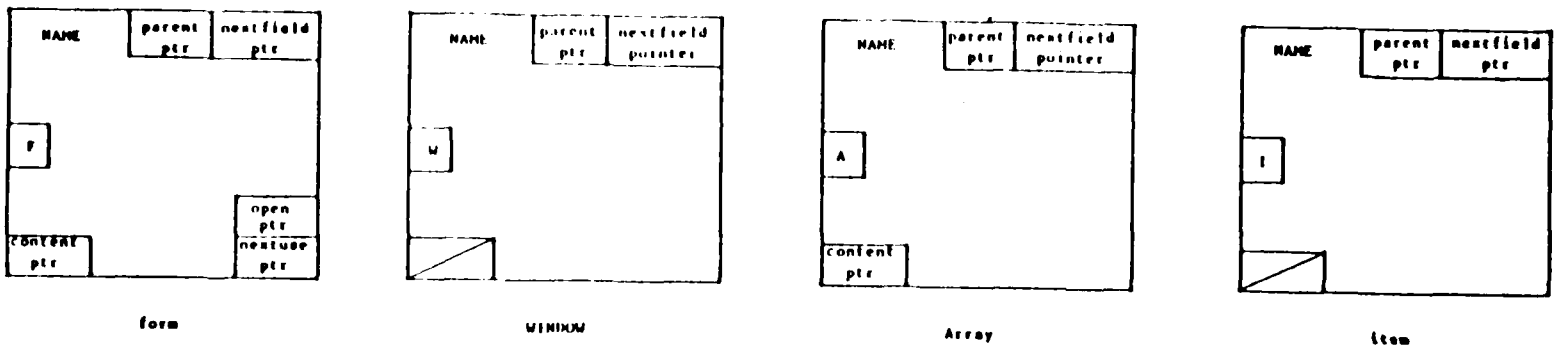
3.2.3.3.3 Application List

This list contains all User Interface applications. Each application list element has associated with it a user and that user's list of applications and a list of logical devices being used by the application.

3.2.3.3.4 Logical Device List

This list contains all User Interface currently opened logical devices. Associated with a logical device list element is an application, a user, an open list, and a display list.

OPEN LIST ELEMENTS



NOTES:

1) Display list elements are the same except a window on the display list may have a content pointer but on the open list it is always null.

2) Open ptr in the open list and display list always links a form used within another form to the original (top level) definition of the form in the open list.

3) Nextuse ptr in the open list and display list chains all of the particular form occurrences through the open list and into the display list.

 signifies Null pointer

Figure 3-6 Open/Display List Elements

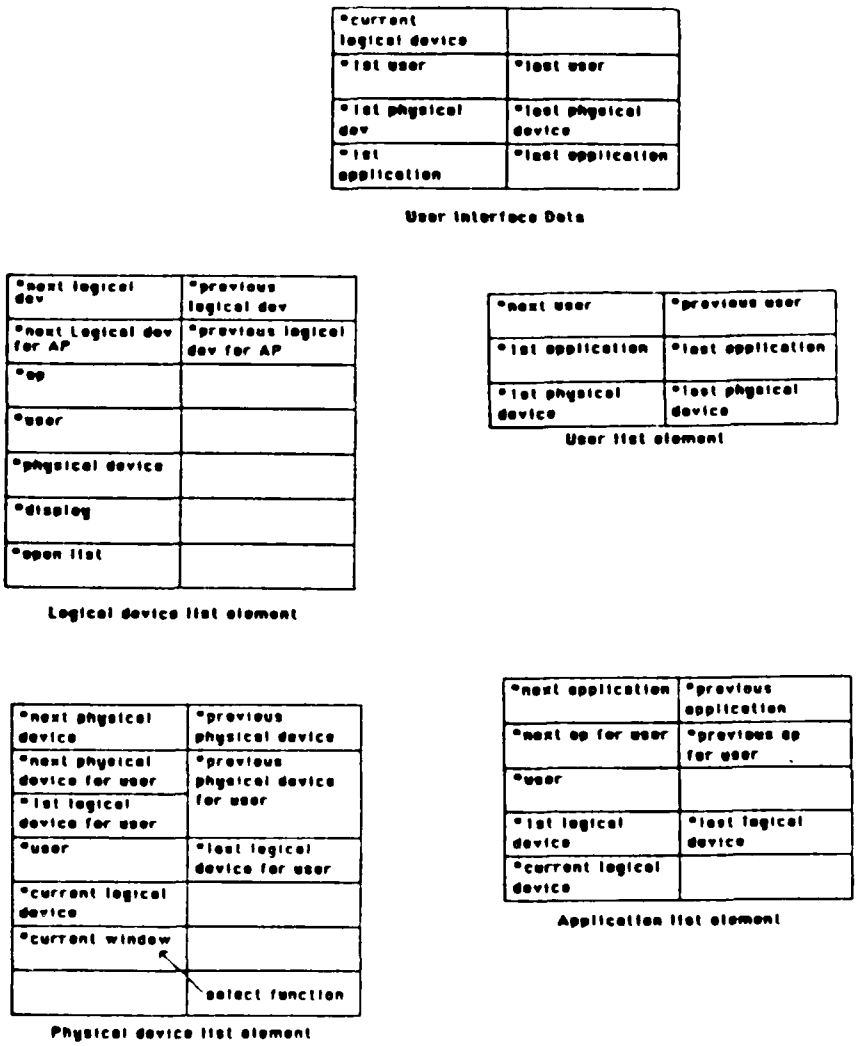


Figure 3-7 Window Management Elements

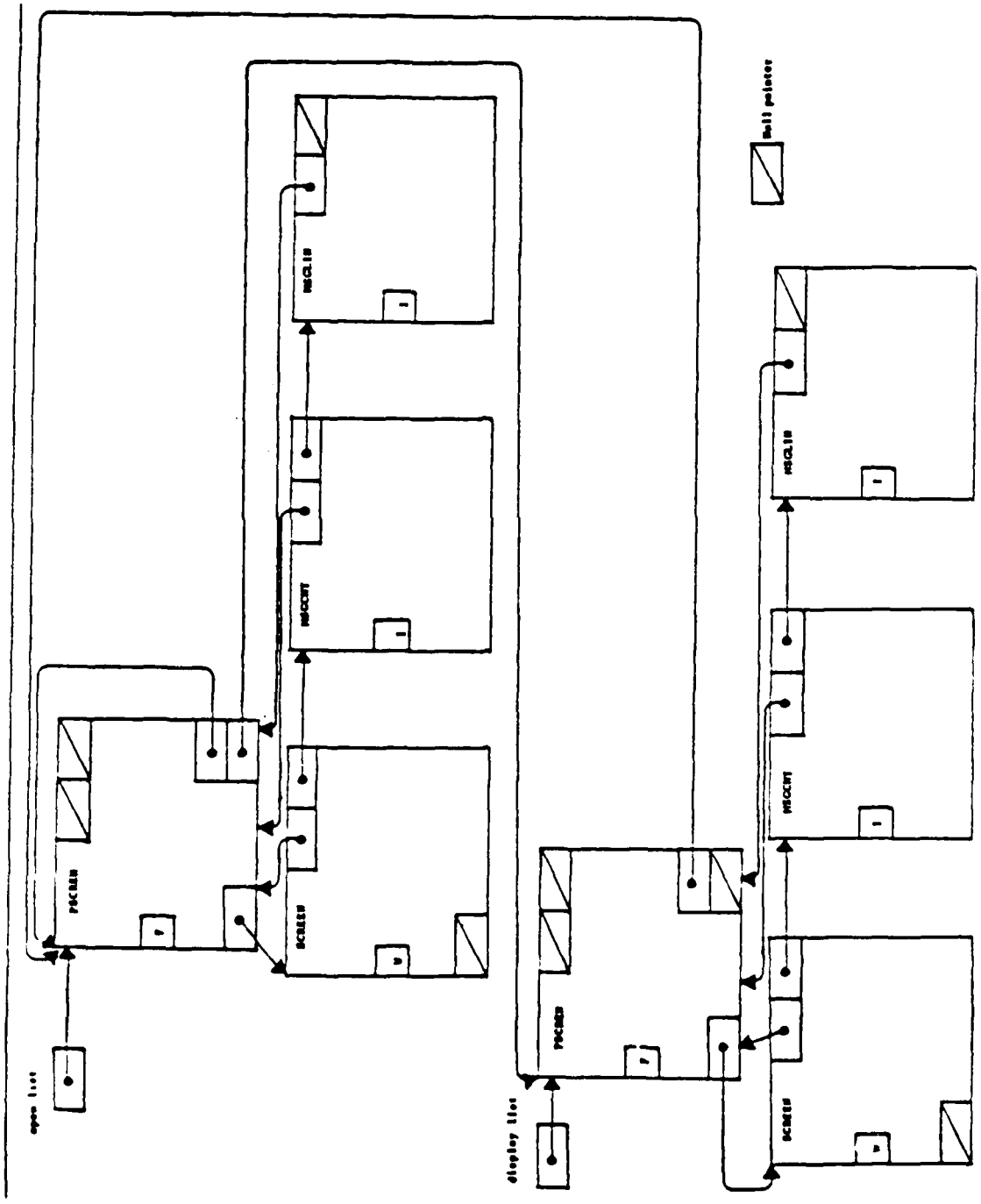


Figure 3-8 Open/Display List After FP Initialization

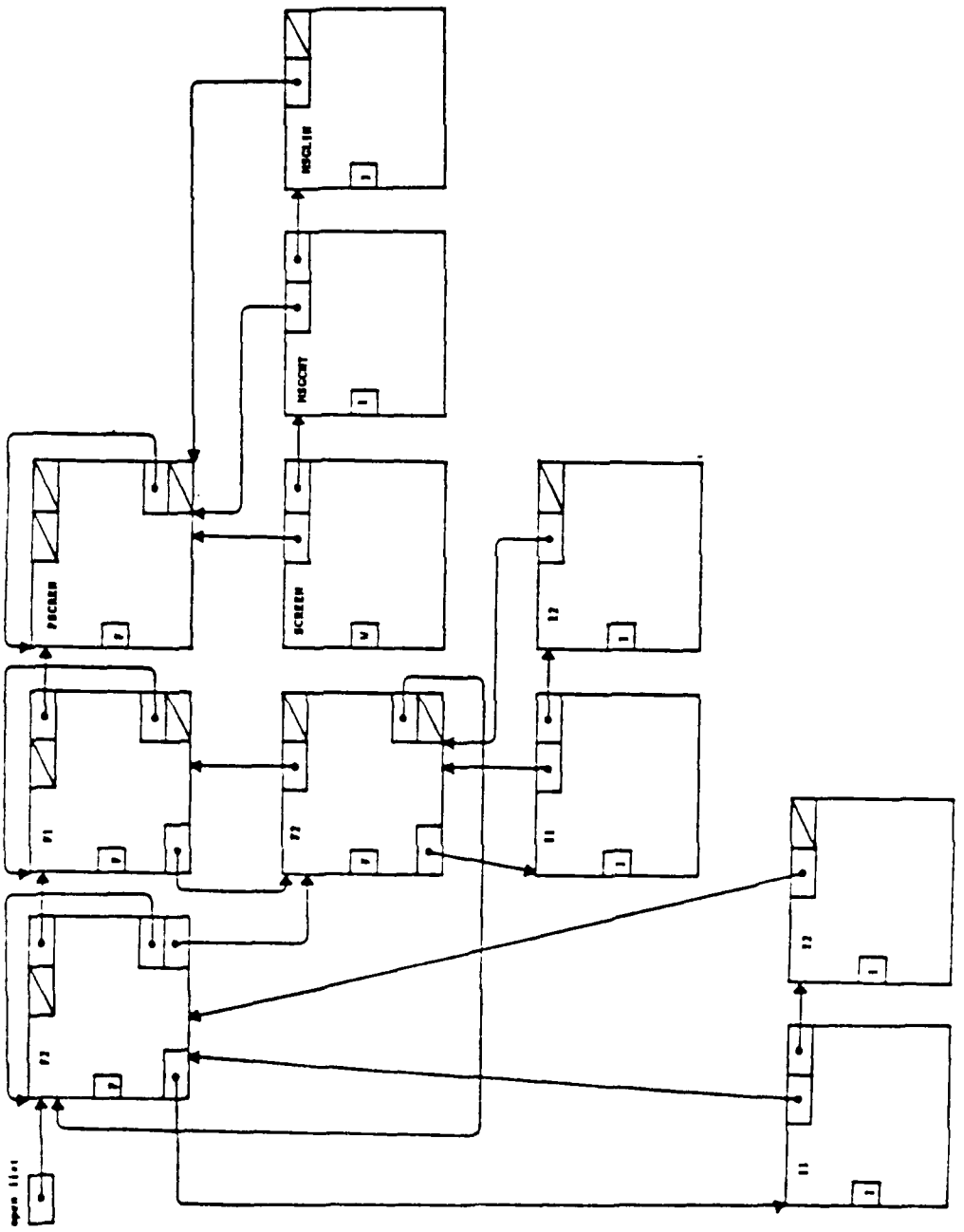


Figure 3-9 Opening a Form F1 Containing Form F2

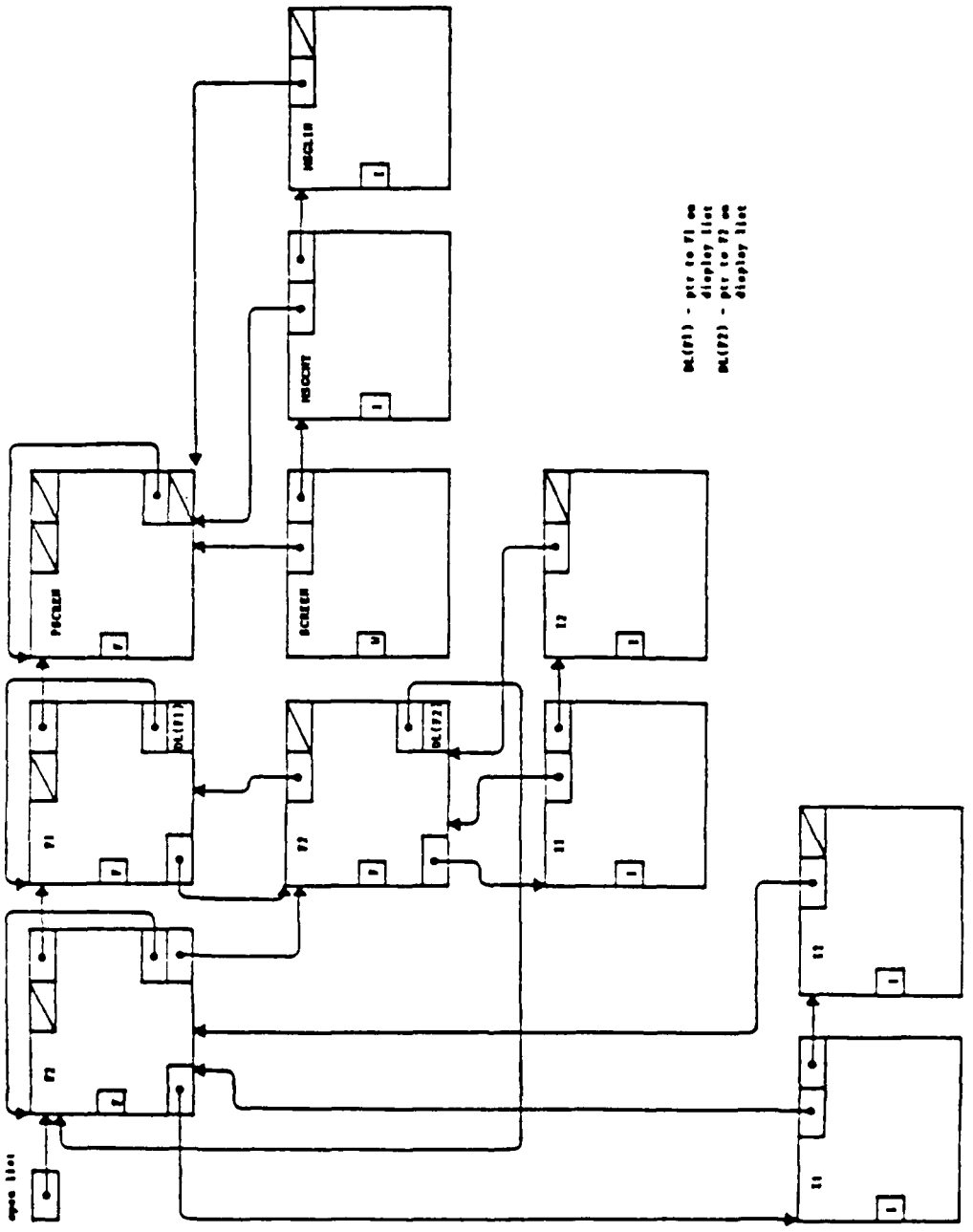


Figure 3-10 Open List After Adding F1 to Window Screen in Display List

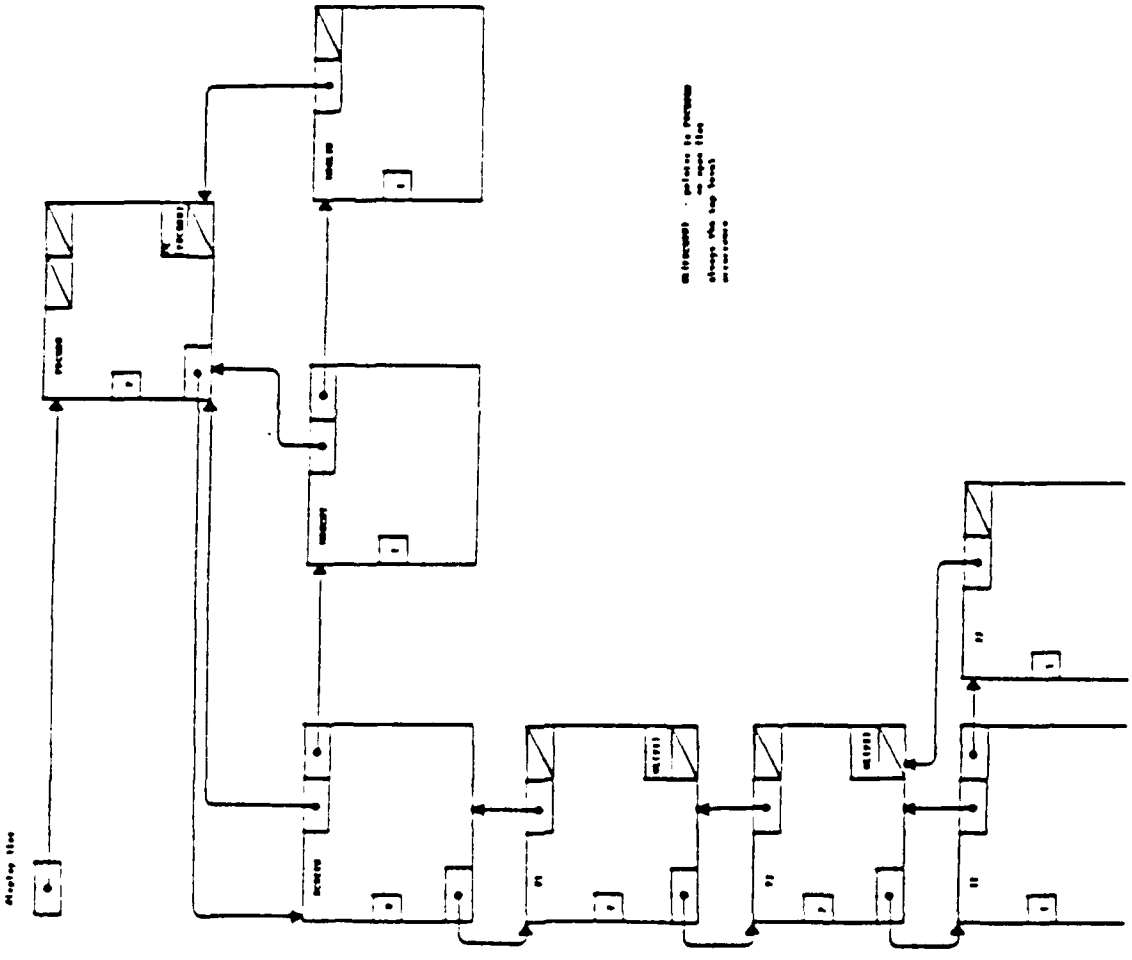


Figure 3-11 Display List After Form F1 Is Added to Screen

3.2.3.4 Internal Data Schema

The following table in Figure 3-12 defines the relationships among the data and the operations on this data from an internal view of the FP functions. The following abbreviations are used in the table:

C - Create
D - Delete
E - Examine
M - May
R - Retrieve
U - Update

Function	OPEN	DISPLAY	LOGICAL	PHYSICAL	APPLICATION
ADDFRM	MC(form)	C(form)			
CHGLDV					U
CLSFRM	D(form)				
CLSLDV			D		
GDATA		R(item data)			
GDATLN		E(item data)			
GETATT		R(item attribute)			
GETBAK		R(form/window attribute)			
GETCUR		E			
GPAGE		R(form name)			
GWINDO		R(window/page #)			
INQLDV			R		R
INITFP	C	C			
OISCR		E			

-CONTINUED-

Function	OPEN	DISPLAY	LOGICAL	PHYSICAL	APPLICATION
OPNFRM	C(form)				
OPNLDV			C		
OUTSCR					
PARFQN					
PDATA		U(item data)			
PMSGLC		U(message)			
PMSGLS		U(message)			
PUTATT		U(item attribute)			
PUTBAK		U(form/window attribute)			
PUTCUR		E			
RMVPAG		D(form)			
RPLFRM		U(form)			
TERMFP	D	D			

Figure 3-12 Table of Internal Data Changes

ERROR MESSAGE MAPPING

	BOATLN	AODPRM	CHBLDV	CLSFRM	CLSIDV	ADAIA	BETATT	BFTBAR	BFTCUR	BPAGF	DWINDO	INITFP	INOLDV	OISCR	DFNFRM	DPNLDV	DUTSCR	POATA	FARFON	PMSBLC	FMSOLS	PUTATT	FUTBAR	PUTCUR	RMVPAG	RPLFRM	TERMFP	
ALCERR	X	X				X	X	X		X	X			X	X		X	X				X	X	X	X	X		Memory Allocation error
AINOTFND							X	X														X	X					Attribute specified does not exist
BLKFLD																												Field must be entered
BSEND	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Bad Send from UI to AP
BUFOVFL														X														VTI Buffer overflow
CURRPOST					X																							Not the Current Logical Device
EDITERR														X														Field does not pass edit checks
ENDARY																												End of scrolling section
FILRERR		X													X											X		File Read ERROR
FILFRMMM		X													X											X		File & form names are mismatched
FINUSE				X																								Form in use, may not close
FISOPEN															X													Form is already open
FNDIFND		X		X											X											X		Form not found
FNDTOPN				X																								Form is not open
FPMSD																												Form processor message
BTCURERR									X																			Could not build qualified name
IMBBLK																												Field must not contain blanks
INOTNUM																												Field must be numeric
INVIID						X																						Invalid instance id
INVPAG										X															X	X		Invalid Page
MAPFAIL														X														VTI Buffer format error
MFOPNERR												O								O								Message file cannot be opened

Figure 3-13 Error Message Mapping, p. 1

	BDATLN	ADDFRM	CHGLDV	CLSFRM	CLSLDV	BDATA	GETATT	GETBAE	BETCUR	GPABE	BMINDO	INITFF	INQLDV	OIBCR	OPNFRM	OPNLDV	OUTBCR	PDATA	PARFON	PMBBLC	PMBBLB	PUTATT	PUTBAE	PUTCUR	RMVPAB	RPLFRM	TERMFP	
MFRDERR												O								O								Encountered read error on message
NFPOSTRC			X		X								X			X												Logical device not open
NOHELP																												No help available
NOHLPFRM																												No help form available
NOTSCROL																												Fields are not scrollable
NTMINT																												Bad return from NTM INITEX routine
NTMLO																												Bad return from NTM Logoff routine
NTMLOG																												Bad return from NTM Logon routine
NTMREJ	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	NTM Reject
NULDNAM																												Null field name
OLDFORM		X													X											X		Old form file recompile
OPNERR		X													X											X		Unable to read FORM definition file
OUTRNG																												Field is out of range
PARSERR																			X									Level non-existent in name
PATHERR	X					X	X	X		X	X			X			X	X				X	X	X	X	X		Invalid path name
PNOTBAK								X															X					Path name is not a window or form
PNOTFND	X					X	X	X		X	X			X			X	X				X	X	X	X	X		Path not found
PNOTITEM							X															X						Pathname is not an item
PNOTUNQ	X	X				X	X	X		X	X			X			X	X				X	X	X	X	X		Path not unique
PNOTWIN		X									X			X			X								X	X		Pathname is not a window
SYSERR														X														System error
UNKTYPE														X														Unknown field type
UOPFRM													O															Bad return from open form routine

X - return in STORE
O - only recorded in error log

Figure 3-13 Error Message Mapping, p. 2

3.2.4 Functional Processing

The calling sequence, description and parameter descriptions found in the Form Processor User Manual describe in detail each routine's input, output, and processing. Figure 3-13 describes the possible error conditions that may result from using the Form Processor routines.

3.3 Special Requirements

3.3.1 Programming Methods

The FP shall be programmed using structured design and care shall be taken to insure portability of the the FP code with minimum effort. Basic programming standards for readability and ease of debugging shall be followed.

3.3.2 Expandibility

Since the FP has been designed as a set of interface routines, new functionality may be added by simply adding new interface routines. The interface between the UIM of the Form Processor and the AI would simply be changed to add a new interface routine which the UIM of the Form Processor would be capable of calling. The operations on the internal data structures: the Open List, the Display List, and the Window Management Lists are abstracted into separate functions which may easily be used by any new interface routines added to the FP

3.4 Human Performance

Performance requirements for the FP have not yet been determined. These requirements should included a statement of response time for displaying and entering a screen and the ease of use of the features such as scrolling/paging and reserved keys.

3.5 Data Base Requirements

The data base requirements listed here are in addition to those outlined in the Detailed Functional Requirements section.

3.5.1 Sources and Types of Input

3.5.1.1 FP Processing Data

The FP requires a data structure containing the current status of processing and also information about the various attributes allowed for the form fields.

The following information is maintained by the FP:

Location of Open List
Location of Display List
Location of Application List
Location of User List
Location of Physical Device List
Current Logical Device
Window Identification Table
NTM Channel Table

3.5.1.2 FDL Form Description File

The FP reads records that are created by the FDL compiler. These records define the forms that are opened by the application program. The FP reads one file per form. This file contains records defining the form version, the form, the text on the form, and the fields within the form. A detailed layout of these records may be found in the Forms Definition Language Compiler Development Specification.

3.5.2 Destinations and Types of Output

The output is fully described by the internal data structures and the processing outlined in the Detailed Functional Requirements section.

3.5.3 Internal Tables and Parameters

The initial attribute definition table is defined at FP initialization. The following attributes are defined:

TEXT
INPUT
OUTPUT
ERROR
BLACK
WHITE
XPARNT (transparent)
HIDDEN
GUARDED

At initialization of the FP, the Attribute Map List is built which is a list of all these valid attributes.

SECTION 4

QUALITY ASSURANCE PROVISIONS

4.1 Introduction and Definitions

The FP is tested using the following tests:

Computer programming test and evaluation. This testing primarily involves testing all the FP interface routines and internal functions for correct processing and output.

System test. This testing involves testing all the FP interface routines and internal functions within the integrated system.

4.2 Computer Programming Test and Evaluation

The test developed for the FP consists of another computer program which uses the FP interfaces routines by an application program. Every FP interface routine is exercised directly by the computer test program and every internal function is indirectly exercised by the computer test program. Variations on the computer test program may be provided by user interaction with the computer test program.

The same computer test program is run to test the FP during the system test process with the other components of the IISS.

SECTION 5

PREPARATION FOR DELIVERY

The implementation site for the constructed software is the ICAM Integrated Support System (IISS) Test Bed site located in Schenectady, NY. The software associated with each FP CPCI release is delivered on a media which is compatible with the IISS Test Bed. The release is clearly identified and includes instructions on procedures to be followed for installation of the release.

APPENDIX A

WINDOW MANAGEMENT MAPPINGS

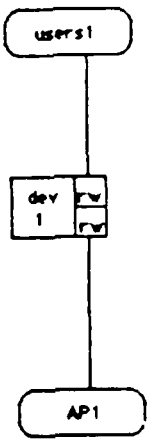


Figure A-1 Single User, Single Device, Single Application

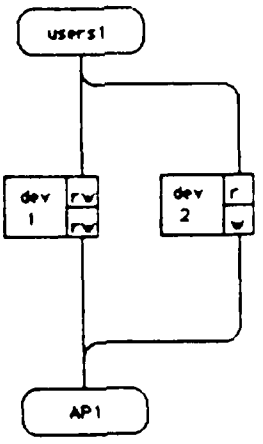


Figure A-2 Single User, Multiple Devices, Single Application

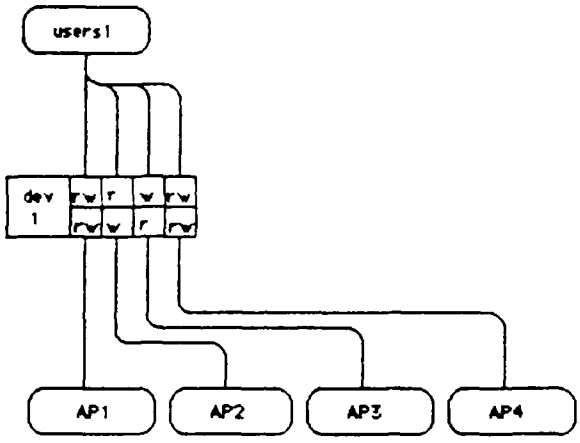


Figure A-3 Single User, Single Device, Multiple Applications

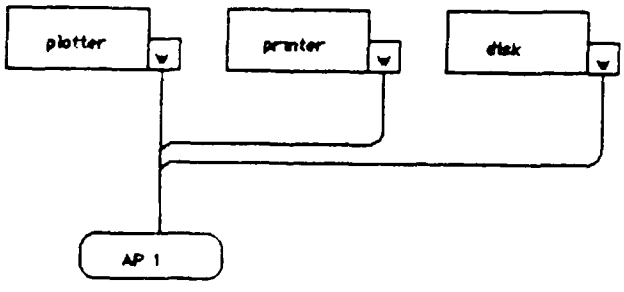


Figure A-4 No User, Single Application, Multiple Devices

APPENDIX B

WINDOW MANAGER INFORMATION FORM

Window Manager

Application	Device Type	Device Name	Pri	Window Name	Location Row	Location Col	Display Size W	Display Size D	Viewport Offset Row	Viewport Offset Col
SDARTESTZZ	SDPRINTZZ	PRINT.DEV	1		6	17	36	11	0	0
SDARTESTZZ				SCREEN	1	1	80	23	0	0
SDARTESTZZ				W3	2	60	10	8	0	0
SDARTESTZZ	VT100	TT:	2		1	1	80	24	0	0
SDARTESTZZ				SCREEN	1	1	80	23	0	0
SDARTESTZZ				W3	2	60	10	8	0	0
SDARTESTZZ				W3	2	60	10	8	0	0
MENU	VT100	TT:	3		1	1	80	24	0	0
MENU				SCREEN	1	1	80	23	0	0

Msg: 0

window mgr

Figure B-1 Window Manager Information Form

END

8-87

DTIC